САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра технологий программирования

Ермолаев Алексей Николаевич

Выпускная квалификационная работа бакалавра

Разработка методов и алгоритмов анализа социальных сетей с использованием теоретико-графовых подходов

Направление 01.03.02

Прикладная математика и информатика

Научный руководитель:
доктор технических наук,
профессор Печников А. А.

Заведующий кафедрой:
кандидат физ.-мат. наук,
доцент Блеканов И. С.

Рецензент:
кандидат физ.-мат. наук,
Чиркова Ю. В.

Санкт-Петербург

2018 г.

Содержание

Введение

Развитие социальных сетей произвело революцию в области социальных графов. Стремительный рост пользовательской базы Facebook и Twitter привёл к резкому падению посещаемости таких первых социальных сетей, как Friendster и Myspace, в связи с чем внимание учёных и исследователей в области социальных графов сфокусировалось главным образом именно на двух новых лидерах рынка.

В настоящий момент зарегистрировано более 2 миллиардов аккаунтов в сети Facebook [1], что и объясняет популярность использования этой площадки для анализа социальных сетей в научных и коммерческих целях. Такая многочисленная и разнообразная аудитория позволяет оптимизировать проблему различия социальных сетей, сконцентрировать внимание на исследовании взаимоотношений, связей, сегментации объектов и выстраивать эффективные модели социальных графов.

Сторонние компании предоставляют инструменты для классификации пользователей по различным признакам, начиная от социальных групп, объектов, сообществ, медиаконтента и заканчивая возрастом, полом, национальностью, общественным положением, уровнем образования и т.д. В свою очередь классификация позволяет определить особенности социальных отношений, выявить пересечения интересов, узловые тематические моменты и, как следствие, направленность социальных векторов. Такая информация несомненно окажется полезной не только при оценке собственной аудитории, но и для выявления различного рода зависимостей в общественных отношениях.

Нельзя забывать, что общение пользователей социальных сетей — это не только отражение общественных отношений, но и динамика происходящих в обществе изменений. Характер противоречий, социальная активность и пассивность, реакция на внешние и внутренние раздражители, принятие решений, политические симпатии и антипатии, анализ и прогноз ситуации – все это при использовании определенных методологических инструментов позволяет определить степень влияния социальных сетей не только на формирование общественных процессов, изменения в обществен-

ном сознании, но и, в конечном счете, на структурирование общества в определенной временной перспективе. Такие моменты, как отсутствие дискретности и социальных барьеров в общении пользователей сетей, скорость распространения информации, более высокая степень независимости суждений и мнений и ряд других лишь способствует объективности исследования.

Разумеется, что процесс анализа особенностей социальных сетей происходит на «стыке» наук, поскольку его результаты представляют несомненный интерес для социологов, статистиков, психологов, политологов, что лишний раз доказывает важность и перспективность предпринимаемых исследований.

По состоянию на лето 2017 года самой популярной социальной сетью в России является Вконтакте [2]. В связи с вышеописанным обуславливается актуальность данной работы. Новизна заключается в исследовании социальных графов, являющихся подграфами специального вида, построенными на подмножествах участников, имеющих одного общего друга (эго-сетей) и сетей друзей и их друзей (эго-эго сетей). Были разработаны инструменты, отсутствующие в свободном доступе [3], для эффективного сбора и анализа таких социальных графов и вычисления их основных метрик для социальной сети ВКонтакте, проведена серия экспериментов и сделан ряд выводов, и позволяющих наметить новые направления исследований.

Постановка задачи

Задача заключается:

- в разработке метода построения социальных графов для социальной сети ВКонтакте специального вида:
  - эго-граф друзей;
  - граф друзей и их друзей;
  - граф пользователей сообщества;
- в разработке программы для вычисления метрик социального графа;
- в проведении серии экспериментов на реальных сетях и обобщении полученных результатов.

Для этих целей будет разработана программа SGMaker с графическим интерфейсом, удовлетворяющая вышеперечисленным требованиям, которая представит практическую ценность для специалистов и исследователей в области социальных сетей, так как упростит и ускорит процесс сбора, обработки и подготовки данных к анализу. Программа должна быть ориентированной на «обычного» аналитика в конкретной предметной области.

Очевидно, что разработка системы интеллектуального анализа социальных сетей, удовлетворяющая потребности широкого спектра исследователей, достаточно сложна, но реализуема посредством затрат большого объёма ресурсов.

Обзор литературы

При написании данной работы автором были использованы различные источники, непосредственно или опосредованно касающиеся её темы: статьи из научных изданий, публикации в интернете, учебная литература.

Вывод об актуальности разработки удобной программы для сбора информации из социальной сети ВКонтакте и построения социальных графов был сделан на основании источников, в которых описаны исследования, в ходе которых были изучены различные подходы, применяющиеся для сбора исходной информации [4], [5], а также содержится обзор современных систем анализа социальных сетей [6].

Необходимые сведения для изучения анализа социальных графов в контексте сложных сетей были получены из [7], [8]. В них даются основные понятия теории сложных сетей и описаны используемые способы изучения существующих сетевых структур.

Общие понятия, история, метрики и методы анализа социальных сетей основаны на информации из [9], [10]. В них формулируется определение социальным сетям интернет-пространства, приводится необходимый набор инструментов и навыков, способствующий их познанию, описываются различия между онлайн-сетями и другими структурами данных. Также приведены конкретные примеры сбора и анализа данных социальных сетей.

Полезными для работы стоит отметить источники [11], [12], где подробно описывается и исследуется такое понятие, как ассортативность или ассортативное смешивание в сетях. Применение этого подхода для исследования социальных сетей позволяет обнаруживать новые структурные свойства сетей. В статье [13] определили скалярную меру ассортативного смешивания и использовали ее, показав, что многие социальные сети имеют значительное ассортативное смешивание.

Так же были изучены исследования других социальных сетей, где анализируются основные количественные характеристики, такие как корреляция при образовании связей, кластеризация вторичных связей и распределение вершин по числу соседей, приводятся выводы о структуре связей в социальной сети Живой Журнал (LiveJournal) [14]. В работе [15]

изучалась социальная структура сетей «дружбы» Facebook в ста американских колледжах и университетах в один момент времени. Был сделан вывод, что вычисление коэффициентов ассортативности и коэффициентов регрессионной модели на основе наблюдаемых связей позволяет исследовать гомофилию на локальном уровне, а алгоритмическое обнаружение сообществ позволяет получить дополняющую макроскопическую картину.

Глава 1. Базовые понятия теории графов

§1.1 Основные определения

Граф или *сеть* — это формальное представление структуры отношений (дружбы, коммуникации, управления и др.).

В данной работе будут рассматриваться простые (неориентированные, невзвешенные, без петель и кратных рёбер) графы $G = (V, E)$, которые состоят из непустого множества вершин V, и множества неупорядоченных пар отдельных элементов из V , называемого множеством рёбер E. В качестве дополнительной информации используются метки «возраст» и «город».

Полный граф — граф, в котором любые две вершины связаны ребром.

Клика — полный подграф.

Степенью вершины v графа $G = (V, E)$ называется число рёбер, инцидентных вершине v.

Признаковая степень вершины v графа $G = (V, E)$ — число смежных вершин с таким же значением признака как у v.

Социальный граф в интернет-контексте — это граф, который отображает социальные связи между социальными объектами. Это модель представления социальной сети, где термин «граф» взято из теории графов. Социальными объектами могут быть сообщества, медиаконтенты и профили пользователей, обладающие определёнными атрибутами: имя, место проживания, возраст, социальный статус.

Эго-граф друзей — структура социального графа, состоящая из центрального узла («эго») и узлов, с которыми эго непосредственно связано (в данной работе — «друзья эго»), а также связи, если таковые имеются, между ними.

Граф друзей и их друзей — это эго-граф друзей, расширенная эго-графами друзей каждого из «друзей» центрального узла.

Граф пользователей собщества — структура социального графа, где узлы представлены пользователями тематического сообщества социальной сети ВКонтакте, а также связи, если таковые имеются, между ними.

§1.2 Метрики

Для анализа особенностей социальных сетей необходим гибкий инструмент, который помогает отобразить числовые характеристики социальных объектов, сегментов, групп и что самое важное — их связей. Таким инструментом являются метрики различных характеристик. В решении задач при изучении социальных графов используются следующие метрики:

- метрики взаимоотношений, отображающие характер взаимоотношений между различными социальными объектами;
- метрики связей, показывающие особенности взаимодействия отдельных социальных объектов и социального графа в целом (центральность, плотность, ассортативность);
- метрики сегментации, показывающие характеристики социального графа, разбитого на определенные сегменты (коэффициент кластеризации).

В работе рассматривались плотность, коэффициент кластеризации и коэффициент ассортативности по признакам «город» и «возраст».

Плотность

Достаточно распространённым для рассматриваемого нами вопроса является понятие плотности. Это отношение числа фактически имеющихся ребер к максимально возможному количеству ребер определенного графа. Таким образом, используя метрику плотности, мы можем не только сравнивать графы одного размера, но и рассматривать дальнейшее развитие графа во времени.

$$d = \frac{2\,|E|}{|V|\,|V-1|}, \tag{1}$$

где d — значение плотности, $|E|$ — мощность множества рёбер, $|V|$ — мощность множества вершин.

Кластеризация

Кластеризация является важным свойством социальных сетей: люди, как правило, «дружат» с теми, кто уже дружит между собой, в результате получаются группы людей с большим количеством рёбер, в то время как набор, сделанный из случайно выбранных людей, будет иметь гораздо меньшее количество ребер. Для измерения кластеризации в социальной (или другой) сети общей мерой является коэффициент кластеризации. Коэффициент кластеризации является действительным числом между нулем, когда нет кластеризации, и единицей, когда кластеризация максимальна. Последнее получается в том случае, когда сеть состоит из непересекающихся клик. В то время как кластеризация в сети может быть измерена несколькими способами, один общий способ сделать это — проверить треугольники. В сети с высокой кластеризацией два ребра, совместно использующих узел, вероятнее всего, завершены третьим, тем самым, образуя треугольник. Существуют два способа вычисления коэффициента кластеризации графа, которые могут иметь совершенно разные значения:

- Первый способ. Глобальный коэффициент кластеризации основан на триплетах узлов. Триплет — это три узла, которые соединены двумя (открытый триплет) или тремя (закрытый триплет) неориентированными связями. Поэтому треугольный граф включает три замкнутых триплета, по одному центрированному на каждом из узлов (это означает, что три триплета в треугольнике происходят из перекрывающихся выборок узлов). Глобальный коэффициент кластеризации — это отношение числа замкнутых триплетов (или трёх треугольников) над общим количеством триплетов (как открытых, так и закрытых).

$$c = \frac{\text{число закрытых триплетов}}{\text{число всех триплетов}},$$

$$c = \frac{3 * \text{число треугольников}}{\text{число всех триплетов}} \tag{2}$$

- Второй способ. Сначала вычисляется локальный коэффициент кластеризации как вероятность того, что два инцидентных ребра будут завершены третьим, формируя треугольник. Два ребра инцидентны, когда

они имеют общую вершину. Коэффициент кластеризации равен доле таких пар инцидентных ребер, которые завершаются третьим ребром для формирования треугольника. Можно сказать, что коэффициент кластеризации узла является мерой того, насколько полной является окрестность узла. Глобальный коэффициент вычисляется как среднее относительно всех локальных.

$$c_v = \frac{2T(v)}{deg(v)(deg(v)-1)}, \tag{3}$$

$$c = \frac{1}{n}\sum_{v \in V} c_v, \tag{4}$$

где c_v, c — соответственно локальный и глобальный коэффициенты кластеризации узла v, $T(v)$ — количество треугольников, к которые входит v, $deg(v)$ — степень узла v, n — мощность множества вершин V графа G.

Первый вариант был предложен Д. Уоттсом и С. Строгацом [16]. Однако в [7] Марк Ньюмен утверждает, что этот метод менее предпочтителен, чем второй, так как в сети, где преобладают узлы с низкими степенями, получается большое значение глобального коэффициента кластеризации, что, по его мнению, нерепрезентативно. В работе использовался второй вариант.

Ассортативность

Известно, что узлы в графе соединены не случайным образом. Например, отдельные лица предпочитают взаимодействовать с другими людьми, близкими по возрасту, религии, образованию [17]. Данное понятие, получившее название «ассортативность», введено Ньюманом [13] в 2002 году и с тех пор широко изучено. Ассортативность характеризует тенденцию смещения в пользу связей между узлами со схожими значениями признака.

Не все «соседи» одного узла одинаково важны. Это является основой, например, индекса центральности Katz [18], PageRank [19] и центральности подграфа [7].

$$k_i = \sum_{j \in N(i)} d_j, \quad d_j = \begin{cases} 1, & s_j = s_i, \\ 0, & \text{иначе.} \end{cases}, \tag{5}$$

где k_i — признаковая степень вершины i, а s_i, s_j — значения признака вершин i, j соответственно.

$$r = \frac{\frac{1}{m}\sum\limits_{(i,j)\in E} k_i k_j - \left(\frac{1}{m}\sum\limits_{(i,j)\in E} \frac{1}{2}\Big(k_i + k_j\Big)\right)^2}{\frac{1}{m}\sum\limits_{(i,j)\in E} \frac{1}{2}\Big(k_i^2 + k_j^2\Big) - \left(\frac{1}{m}\sum\limits_{(i,j)\in E} \frac{1}{2}\Big(k_i + k_j\Big)\right)^2}, \tag{6}$$

где k_i, k_j — признаковые степени обоих концов ребра $(i, j) \in E$, а m — это число рёбер в графе. Положительное значение индекса ассортативности $r > 0$ демонстрирует тенденцию смещения в пользу связей между узлами со схожими значениями признака, а $r < 0$ свидетельствует об обратном [20, 21].

Коэффициент ассортативности — это коэфффициент корреляции Пирсона между степенями соседних узлов. Представляет собой вещественное число в отрезке от -1 до 1.

Глава 2. Программные средства

§2.1 Используемые методы и инструменты

При разработке программы SGMaker, отвечающей поставленным задачам, а также для тестирования и визуализации некоторых результатов использовались следующие программные средства:

- Python 3.6 — интерпретируемый язык программирования высокого уровня для программирования общего назначения. Обладает большим набором полезных функций в стандартной библиотеке, что позволяет повысить эффективность разработки и читаемость кода;

- NetworkX — библиотека Python для создания, обработки и изучения структуры, динамики и функций сложных сетей. Имеет множество готовых алгоритмов для анализа графовых структур;

- vk_requsts — библиотека Python для работы с VK API. Используется для аутентификации и создания сессии во время работы, чтобы избавиться от необходимости авторизации при каждом запросе;

- VK API ВКонтакте — это интерфейс, позволяющий получать необходимую информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Интерфейс прикладного программирования (API) представляет собой набор определений подпрограмм, протоколов и инструментов для создания прикладного программного обеспечения. В общих чертах, это набор четко определенных методов коммуникации между различными программными компонентами;

- Gephi — представляет собой интерактивную платформу визуализации и исследования для всех видов сетей и сложных систем, динамических и иерархических графов. Это дополнительный инструмент к традиционной статистике, поскольку визуальное мышление с использованием интерактивных интерфейсов позволяет облегчить рассуждения и интуитивно обнаруживать закономерности;

- PyQt — это привязки графического программного обеспечения Qt для Python, реализованные в виде модуля Python. Позволяет упростить создание графического интерфеса пользователя (GUI).

§2.2 Описание функций SGMaker

Json-ответ представляет собой результат выполнения определённой функции в формате JSON (англ. JavaScript Object Notation), текстовом формате обмена данными, основанном на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Обозначение id является идентификатором пользователя социальной сети или именем тематической группы.

В SGMaker реализованы следующие функции:

- users_get_id — возвращает json-ответ с id, именем и фамилией запрашиваемого пользователя;
- users_get_city_bdate — возвращает json-ответ id, именем, фамилией, городом проживания и датой рождения запрашиваемого пользователя;
- friends_get — возвращает список друзей запрашиваемого пользователя;
- groups_get_members — возвращает список пользователей запрашиваемого сообщества;
- calculate_age — возвращает возраст по указаной дате рождения, если недостаточно данных вернёт 'n/a';
- download_and_create_graph — основная функция, собирающая данные, строящая социальный граф и сохраняющая результат в директорию
  '\graphs';
- assortativity_age — возвращает коэффициент ассортативности по возрасту для заданного графа;
- assortativity_city — возвращает коэффициент ассортативности по городу для заданного графа;
- density — возвращает плотность заданного графа;
- MainWindow — класс для создания графического интерфейса и обработки действий в нём.

§2.3 Руководство пользователя

Воспользоваться программой можно двумя способами:

- Распаковать архив, доступный по ссылке `https://yadi.sk/d/newHOg-03WQJHv`, и запустить файл-скрипт SGMaker.py в папке SGMaker_script.

  Для этого варианта необходимо наличие установленного Python 3.6 и всех необходимых библиотек (См. Приложения).

- Распаковать архив, доступный по ссылке `https://yadi.sk/d/Je-9DDFt3WQXU8`, и запустить исполняемый файл SGMaker.exe в папке SGMaker.

  Для этого варианта может потребоваться отключение SmartScreen в Windows 10 (См. `https://remontka.pro/smartscreen-off-windows-10/`).

Рекомедуемые системные требования для корректной работы SGMaker:

- Операционная система Windows 7/8/10
- Оперативная память $\sim$ 2 ГБ
- Процессор с двумя и более физическими ядрами
- Свободное место на жёстком диске $\sim$ 1 ГБ.
- Скорость интернет-соединения $\sim$ 256 Кбит/с.

Работа с программой SGMaker:

После запуска программы любым из вариантов откроется интерфейс командной строки и окно приложения.

Для того, чтобы построить граф пользователя или сообщества, нужно:

- В поле *Id* — ввести короткое имя (или Id) объекта, для которого строится граф (См. Рис. 1)
- В поле *Type* — выбрать желаемый тип графа (См. Рис. 2)

- В поле *Save as* — ввести имя, под которым сохранятся два файла в директории приложения в папке \graphs в формате Gephi age.gexf и city.gexf. В файле age.gexf сохранится структура графа с пометками узлов «возраст», а в файле city.gexf – с пометками «город».

- Нажать кнопку *Download graph*

- Дождаться надписи *Done!* или *Completed!* (См. Рис. 3) в консоли, после чего граф будет успешно сохранён на диске. При вводе id несуществующего или деактивированного объекта в приложении справа появится надпись *Invalid data!*.

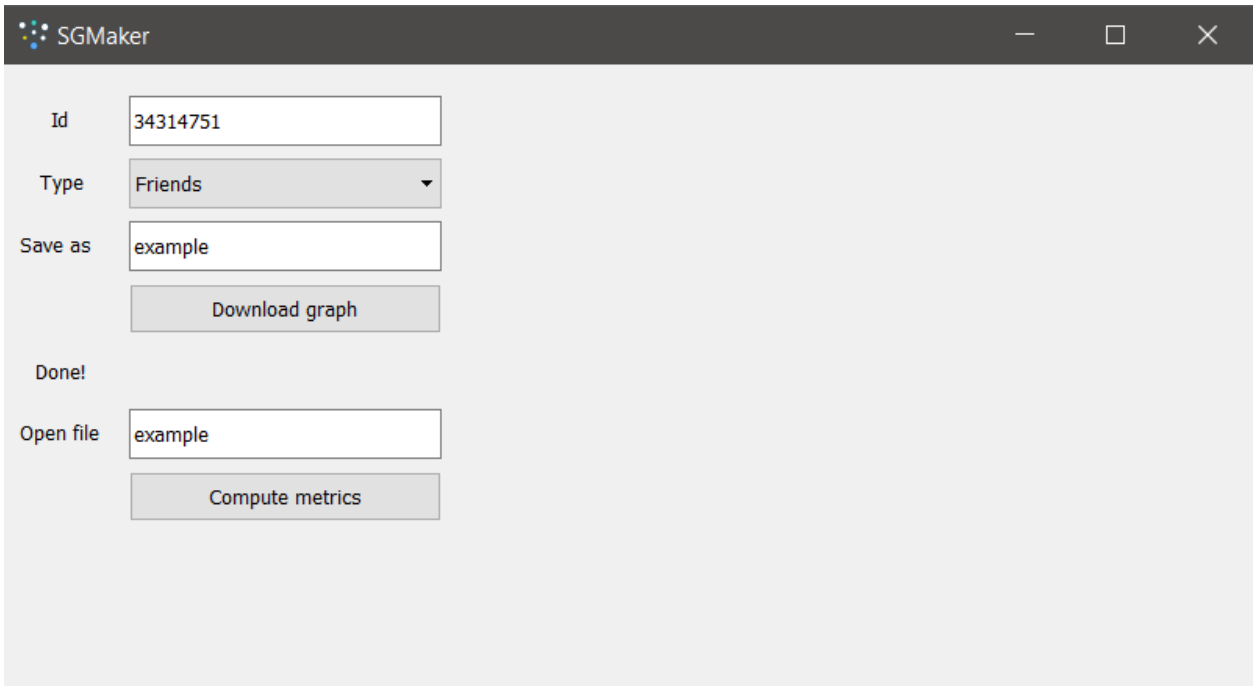

Рис. 1: Пример работы с интерфейсом в варианте Friends

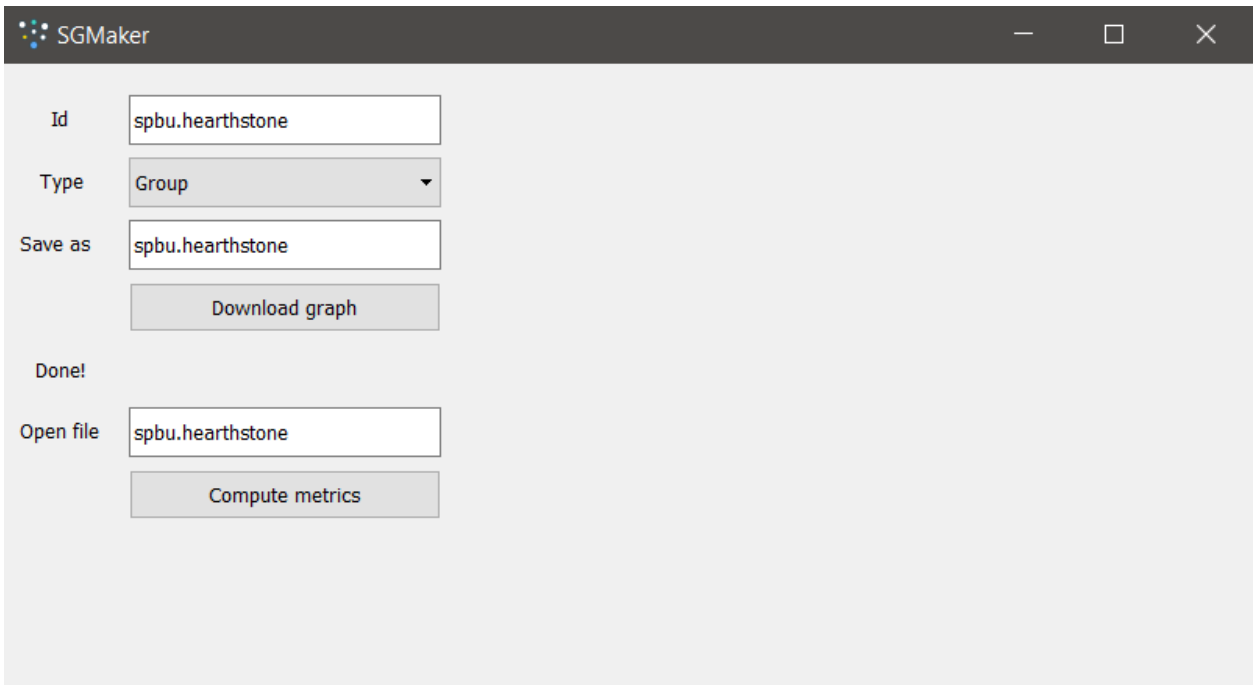

Рис. 2: Пример работы с интерфейсом в варианте Group

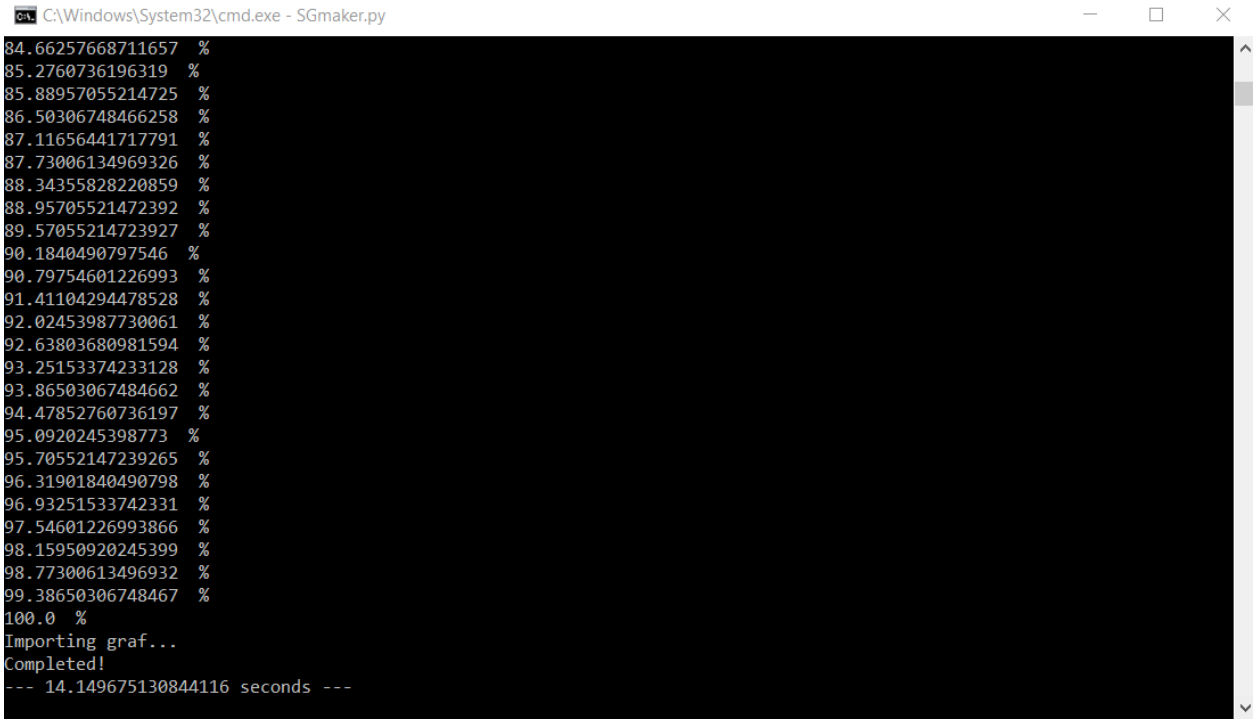

Рис. 3: Пример работы консоли

Для того, чтобы рассчитать метрики имеющегося на диске графа, нужно:

- Если до этого шага выполнялось скачивание графа, то достаточно нажать на *Compute metrics* и в правой части появится информация о количестве узлов, рёбер, плотности, коэффициент кластеризации, коэффициент ассортативности по возрасту и городу графа (См. Рис. 4)
- Для скачанных заранее графов рассчитать метрики можно, указав в поле *Open file* название существующего графа, сохраненного в поддиректории graphs. В противном случае появится надпись *Invalid file name*.

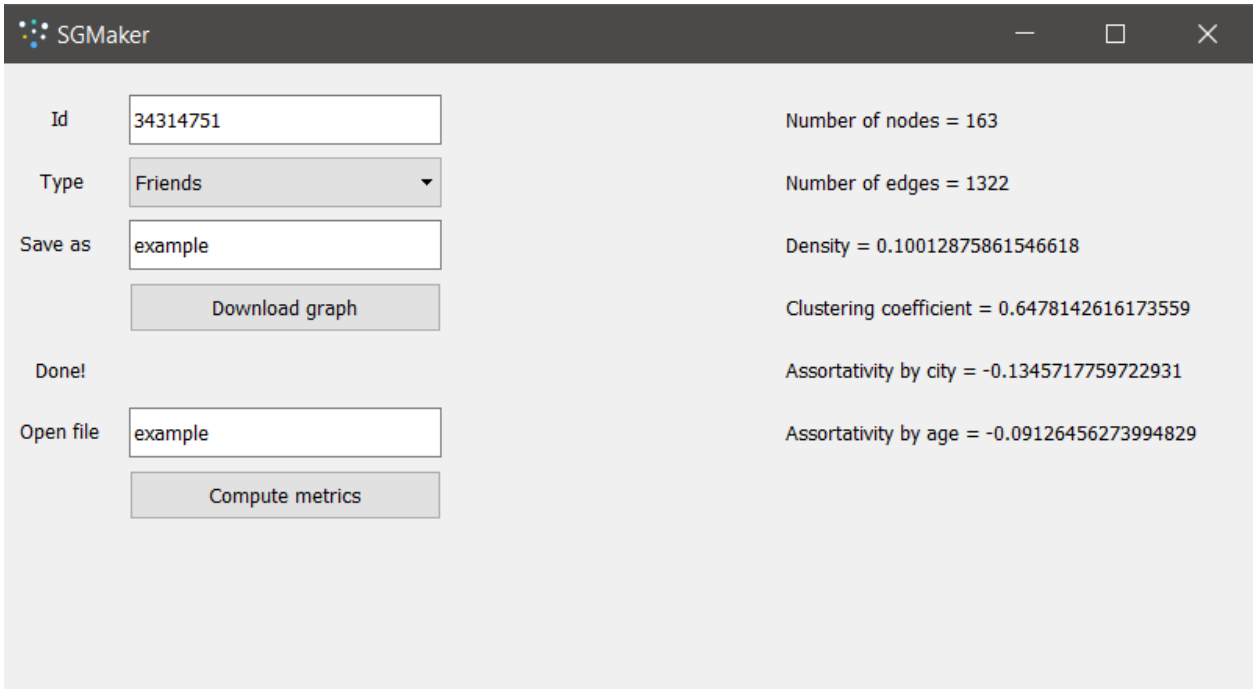

Рис. 4: Пример вывода результатов в SGMaker.

Глава 3. Эксперименты на реальных данных

С помощью разработанного метода построения социальных графов и программы SGMaker был проведён ряд экспериментов на реальных данных.

В качестве примера здесь приведем результаты исследований эго-графа автора выпускной квалификационной работы (id=34314751). Эго-граф в данном случае содержит 159 вершин и 1322 рёбер. Для каждой из вершин были построены свои эго-графы и приводятся обобщённые значения полученных результатов вычисления метрик.

Визуализирован эго-граф автора с помощью Gephi (См. Рис. 5)

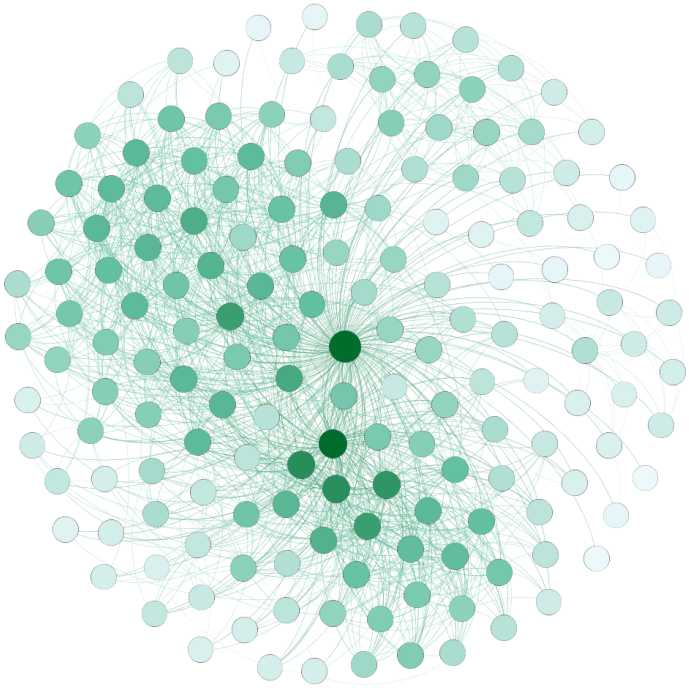

Рис. 5: Эго-граф друзей автора

Здесь вершина, соответствующая автору, находится в центре и выделена наиболее темным цветом. Цвета других вершин соответствуют их степени.

Исследованы 159 эго-графов друзей автора, приведены графики распределения метрик. Эго-графы друзей автора имеют от 12 до 2548 вершин и от 13 до 152695 ребер. «Средний» (хотя такое понятие здесь неприменимо, поэтому взято в кавычки) эго-граф имеет 296 вершин и 3670 рёбер.

Распределение плотности (См. Рис. 6).

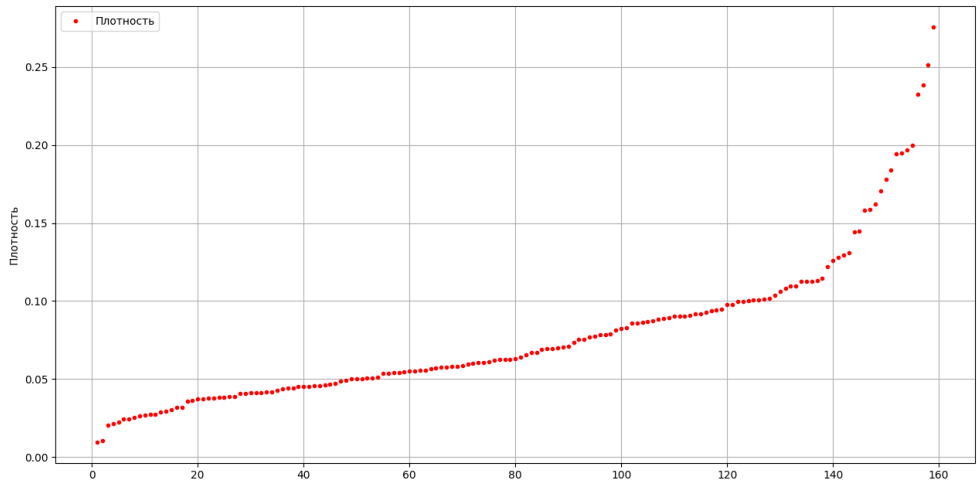

Рис. 6: Распределение плотности

Список значений плотности 159 пользователей был отсортирован по возрастанию и представлен в виде графика распределения. По вертикальной оси отложены значения плотности, по горизонтальной — порядковый номер пользователя из списка.

Распределение коэффициента кластеризации (См. Рис. 7).

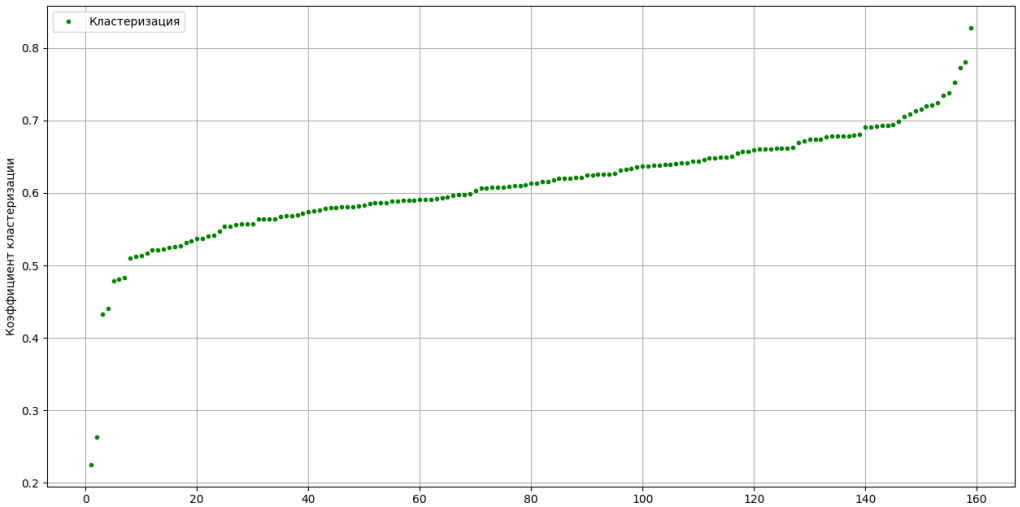

Рис. 7: Распределение коэффициента кластеризации.

Список значений коэффициента кластеризации 159 пользователей был отсортирован по возрастанию и представлен в виде графика распределения. По вертикальной оси отложены значения коэффициента кластреризации, по горизонтальной — порядковый номер пользователя из списка.

Распределение коэффициента ассортативности по возрасту (См. Рис. 8).

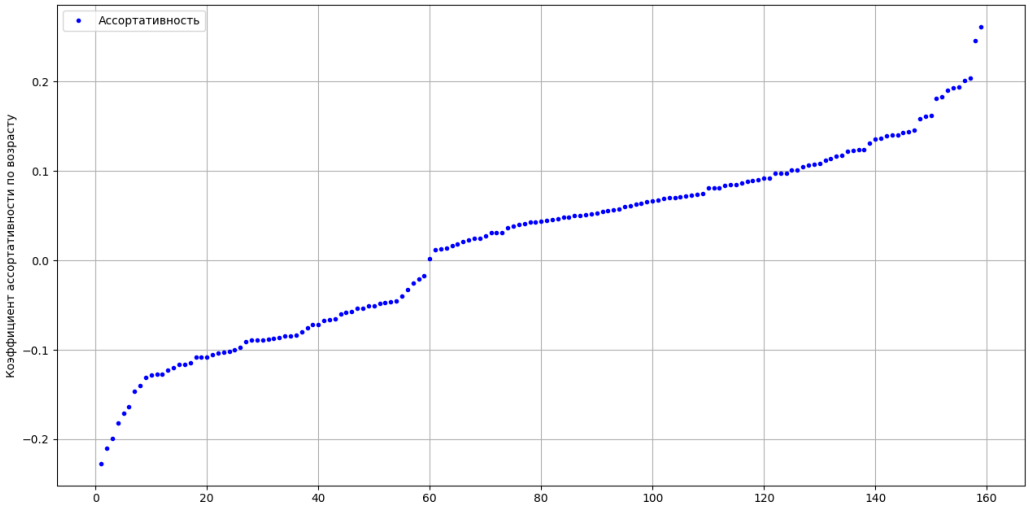

Рис. 8: Распределение коэффициента ассортативности по возрасту.

Список значений коэффициента ассортативности по возрасту 159 пользователей был отсортирован по возрастанию и представлен в виде графика распределения. По вертикальной оси отложены значения коэффициента ассортативности по возрасту, по горизонтальной — порядковый номер пользователя из списка.

Распределение коэффициента ассортативности по городу (См. Рис. 9).

Список значений коэффициента ассортативности по городу 159 пользователей был отсортирован по возрастанию и представлен в виде графика распределения. По вертикальной оси отложены значения коэффициента ассортативности по городу, по горизонтальной — порядковый номер пользователя из списка.

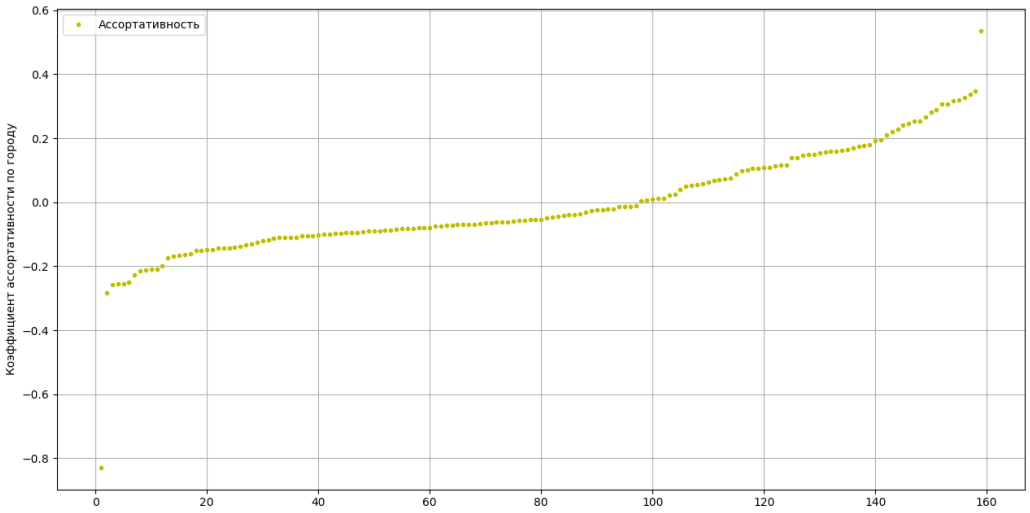

Рис. 9: Распределение коэффициента ассортативности по городу.

Составлена таблица минимальных, максимальных и «средних» значений метрик 159 эго-графов друзей автора (См. Табл. 1).

Таблица 1: Крайние значения метрик 159 пользователей.

	min	max	average
плотность	0.00967	0.27563	0.07815
коэффициент кластеризации	0.22525	0.82792	0.61086
коэффициент ассортативности по возрасту	-0.22711	0.26136	0.02049
коэффициент ассортативности по городу	-0.82868	0.53575	-0.00566

В среднем данные социальные графы обладают низкой плотностью, коэффициентом кластеризации вышего его среднего значения и коэффициентами ассортативности бизкими к нулю. Последнее неудивительно, так как в большинстве — это эго-графы иногородних студентов, живущих в общежитиях.

Приведена гистограмма распределения количества клик в эго-сети автора от их размера (См. Рис. 10).

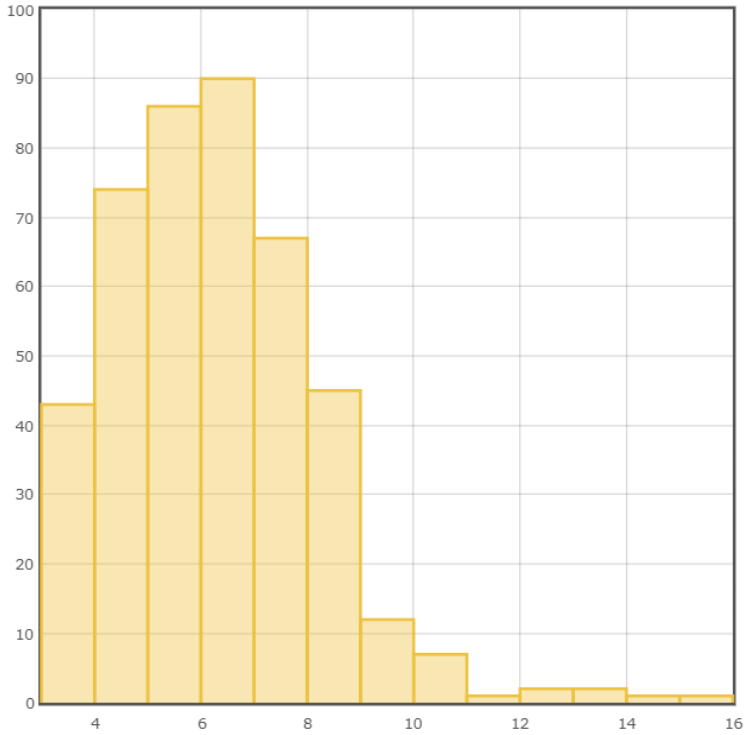

Рис. 10: Распределение количества клик в эго-сети автора от их размера.

Здесь по вертикальной оси указано количество клик, по горизонтальной — их размерность. Наибольшее количество полных подграфов состоят из 6–7 вершин.

Выводы

В процессе достижения целей данной выпускной квалификационной работы посредством реализации программы SGMaker были выполнены все поставленные задачи, а именно:

- разработан метод построения социальных графов для социальной сети ВКонтакте специального вида:
  - эго-граф друзей;
  - граф друзей и их друзей;
  - граф пользователей сообщества;
- разработана программа для вычисления метрик социального графа;
- проведена серия экспериментов на реальных сетях и обобщены полученные результаты.

Исходя из полученных результатов анализа 159 эго-сетей друзей автора, можно сделать вывод о том, что в среднем данные социальные графы имеют низкую плотность ~ 0.08, при этом высокий коэффициент кластеризации ~ 0.6, что означает, что пользователи склонны объединяться в небольшие, но тесно связанные группы. Обращая внимание на распределение клик в эго-сети автора, видно, что пользователи предпочитают «дружить» по 6-7 человек.

Направления для дальнейшей работы и исследований:

- Более подробно исследовать социальные графы на характерные зависимости закономерности в структуре;
- Расширить функционал программы для будущих нужд;
- Ускорить время работы программы;
- Увеличить объём обрабатываемых данных.

Заключение

Алгоритм, реализующий совокупность методов сбора информации, построения моделей социальных графов и вычисления метрик, представляет собой полезный и интересный инструмент не только для пользователей социальных сетей, но и для специалистов в самых различных областях. Результаты, полученные с помощью разработанной программы и методов, могут быть интерпретированы как важный материал для социологов, психологов, политологов. Математическая составляющая функционирования социальных сетей, выявление различных закономерностей возникновения групп и сообществ интернета, несомненно, поможет решению актуальных вопросов и проблем в современном обществе.

Список литературы

[1] 28 Powerful Facebook Stats Your Brand Can't Ignore in 2018 [Электронный ресурс] — Режим доступа: https://sproutsocial.com/insights/facebook-stats-for-marketers/.

[2] Социальные сети в России, лето 2017: цифры и тренды [Электронный ресурс] — Режим доступа: http://blog.br-analytics.ru/sotsialnye-seti-v-rossii-leto-2017-tsifry-i-trendy/.

[3] Мониторинг социальных медиа: 19 эффективных сервисов [Электронный ресурс] — Режим доступа: http://www.topobzor.com/20-servisov-dlya-monitoringa-socialnyx-media/.html.

[4] Коршунов А., Белобородов И., Бузун Н., и др. Анализ социальных сетей: методы и приложения // Труды Института системного программирования РАН. 2014. Т. 26. № 1. С. 439–456.

[5] Яковлев Е. А. Методика и приложения для анализа социальных данных // Научные исследования и разработки 2018 года. 2018. С. 172–186.

[6] Базенков Н. И., Губанов Д. А. Обзор информационных систем анализа социальных сетей // Управление большими системами. Выпуск 41. М.: ИПУ РАН. 2013. С. 357–394.

[7] Newman M. E. J. Networks: An Introduction. Oxford: Oxford University Press. 2010. 1042 p.

[8] Евин И. А. Введение в теорию сложных сетей // Компьютерные исследования и моделирование. 2010. Т. 2 № 2. С. 121–141.

[9] Анализ социальных сетей [Электронный ресурс] – Режим доступа: http://letopisi.org/index.php/Анализ_социальных_сетей/.

[10] Анализ социальных сетей в интернете [Электронный ресурс] – Режим доступа: https://postnauka.ru/longreads/20259.

[11] Newman M. E. J. Mixing patterns in networks // Physical Review E. 2003. Vol. 67. P. 1–13.

[12] Печников А. А. Ассортативное смешивание в российском академическом Вебе //НТИ. Сер. 2. ИНФОРМ. ПРОЦЕССЫ И СИСТЕМЫ 2018. № 1. 2018. С. 8–13.

[13] Newman M. E. J. Assortative mixing in networks // Physical Review Letters. – 2002. Vol. 89. P. 1–5.

[14] Митин Н.А., Подлазов А.В., Щетинина Д.П. Исследование сетевых свойств Живого журнала // Препринты ИПМ им. М.В.Келдыша. 2012. № 78. 16 с.

[15] Traud A., Mucha P., Porter, M. Social Structure of Facebook Networks // Physica A: Statistical Mechanics and its Applications. 2012. Vol. 391. P. 4165–4180.

[16] Strogatz S., Watts D. Collective dynamics of «small-world» networks // Nature | News & Views. 1998. Vol. 393 P. 440–442.

[17] McPherson M., Smith-Lovin L., Cook J. Birds of a feather: homophily in social networks // Annual Review of Sociology. 2001. Vol. 27. P. 415–444.

[18] Katz L. A. New status index derived from sociometric analysis // Psychometrika. 1953. Vol. 18. P. 39–43.

[19] Brin S., Page L. The anatomy of a large-scale hypertextual web search engine // Computer Networks and ISDN Systems. 1998. Vol. 30. P. 1–20.

[20] Allen-Perkins A., Pastor J. M., Estrada E. Two-walks degree assortativity in graphs and networks // Applied Mathematics and Computation. 2017. Vol. 255. P. 1–15.

[21] Estrada E. Combinatorial study of degree assortativity in networks // Physical Review E. 2011. Vol. 84. P. 1–4.

Приложение

Листинг программы SGMaker:

```
import sys
import traceback
from PyQt5 import QtCore, QtWidgets
from PyQt5.QtWidgets import QMainWindow, QLabel, QLineEdit, QComboBox,
    QPushButton
from PyQt5.QtCore import QSize, QRect
from PyQt5.QtGui import QIcon
import networkx as nx
import time
import collections
import vk_requests
from datetime import datetime, date


def users_get_id(function, user_id):
    try:
        t = function(user_ids=user_id, fields=id)
        return t
    except Exception as e:
        if type(e).__name__ == 'VkAPIError' and e.message != 'Access denied:
            user deactivated':
            print(traceback.format_exc())
            users_get_id(function, user_id)
        else:
            print(e.message)


def users_get_city_bdate(function, user_id):
    try:
        t = function(user_ids=user_id, fields=['bdate', 'city'])
        return t
    except Exception as e:
        if type(e).__name__ == 'VkAPIError' and e.message != 'Access denied:
            user deactivated':
            users_get_city_bdate(function, user_id)
        else:
            print(e.message)


def friends_get(function, user_id):
    try:
        t = function(user_id=user_id)
        return t
```

```
    except Exception as e:
        if type(e).__name__ == 'VkAPIError' and e.message != 'Access denied:
            user deactivated':

            time.sleep(1)
            friends_get(function, user_id)
        else:
            print(e.message)


def groups_get_members(function, group_id):
    try:
        t = function(group_id=group_id)
        return t
    except Exception as e:
        if type(e).__name__ == 'VkAPIError':
            groups_get_members(function, group_id)
        else:
            print(type(e).__name__)


def calculate_age(born):
    born = born.split(".")
    today = date.today()
    if len(born) == 3:
        return str(today.year - int(born[2]) - ((today.month, today.day) < (
            int(born[1]), int(born[0]))))
    else:
        return 'n/a'


def download_and_create_graph(g_type, id, file):
    print(id)
    deleted = []
    api = vk_requests.create_api(app_id=6406248, api_version=5.74,
                                 service_token='
                                    be2cdd26be2cdd26be2cdd2659be4d1d4e
----------------------------------------------------------------------
    bbe2cbe2cdd26e48a16a65b1ae99d0374b0a7')

    it = 1
    print("Creating graf...")
    graph = {}
    print("Collecting friend's friends...")

    if g_type == 'Friends':
```

```
    print(users_get_id(api.users.get, id))
    id = users_get_id(api.users.get, id)[0]['id']
    friend_ids = friends_get(api.friends.get, id)['items']
    extended_friends_ids = set(friend_ids)
    extended_friends_ids.add(id)
elif g_type == "Friend's friends":
    id = users_get_id(api.users.get, id)[0]['id']
    friend_ids = friends_get(api.friends.get, id)['items']
    extended_friends_ids = set(friend_ids)
    extended_friends_ids.add(id)
    for i in friend_ids:
        try:
            t = friends_get(api.friends.get, i)['items']
            for j in t:
                extended_friends_ids.add(j)
            print("Added friends from ", i)
        except Exception:
            continue

elif g_type == 'Group':
    group_ids = groups_get_members(api.groups.getMembers, id)['items']
    extended_friends_ids = set(group_ids)
else:
    raise Exception

for i, friend in enumerate(extended_friends_ids):
    print(it / len(extended_friends_ids) * 100, " %")

    try:
        graph[friend] = friends_get(api.friends.get, friend)['items']
    except Exception as e:
        deleted.append(friend)
    it += 1
it = 1
print('Deleting deactivated users:')
for friend in deleted:
    print(friend)
    extended_friends_ids.remove(friend)

print("Adding nodes and edges...")

g1 = nx.Graph(directed=False)
g2 = nx.Graph(directed=False)

for i in graph:
```

```
            try:
                label = users_get_city_bdate(api.users.get, i)[0]
                if 'city' in label:
                    city = label['city']['title']
                else:
                    city = 'n/a'

                if 'bdate' in label:
                    age = calculate_age(label['bdate'])
                else:
                    age = 'n/a'

            except Exception as e:
                city = 'n/a'
                age = 'n/a'
                print(type(e).__name__)
                pass

            g1.add_node(i, label=city)
            g2.add_node(i, label=age)
            print(it/len(graph)*100, "%")
            it += 1

            try:
                for j in graph.get(i):
                    if i != j and j in extended_friends_ids:
                        g1.add_edge(i, j)
                        g2.add_edge(i, j)
            except Exception as e:
                print(type(e).__name__)
                pass

    print("Importing graf...")
    nx.write_gexf(g1, 'graphs\\' + file + '_city' + '.gexf')
    nx.write_gexf(g2, 'graphs\\' + file + '_age' + '.gexf')
    print('Completed!')


def assortativity_age(graph):
    nodes_age = {}
    for node in graph:
        cur_age = graph._node[str(node)]['label']
        if str(node) in nodes_age:
            continue
        if cur_age == 'n/a':
            nodes_age[str(node)] = -1
```

```
            else:
                nodes_age[str(node)] = 0

            for edge in graph.edges:
                if node in edge:
                    if edge[0] == node:
                        adj_node = edge[1]
                    else:
                        adj_node = edge[0]
                    adj_node_age = graph._node[str(adj_node)]['label'].split(';')
                        [0]
                    if cur_age != 'n/a' and adj_node_age != 'n/a':
                        if abs(int(cur_age) - int(adj_node_age)) < 5:
                            nodes_age[str(node)] += 1

        m = len(graph.edges)
        sum1, sum2, sum3 = 0, 0, 0
        for edge in graph.edges:
            k1 = nodes_age[str(edge[0])]
            k2 = nodes_age[str(edge[1])]
            if k1 == 'n/a' or k2 == 'n/a':
                m -= 1
                continue
            else:
                k1 = int(k1)
                k2 = int(k2)
            sum1 += k1 * k2
            sum2 += 1 / 2 * (k1 + k2)
            sum3 += 1 / 2 * (k1 ** 2 + k2 ** 2)

        r_age = (1 / m * sum1 - (1 / m * sum2) ** 2) / (1 / m * sum3 - (1 / m *
            sum2) ** 2)
        print('Assortativity by age = ', r_age)
        return r_age


    def assortativity_city(graph):
        nodes_city = {}
        for node in graph:
            if 'label' not in graph._node[str(node)]:
                continue
            cur_city = graph._node[str(node)]['label']

            if cur_city == 'n/a':
                nodes_city[str(node)] = -1
            else:
```

```
            nodes_city[str(node)] = 0
        for edge in graph.edges:
            if node in edge:
                if edge[0] == node:
                    adj_node = edge[1]
                else:
                    adj_node = edge[0]
                if 'label' in graph._node[str(adj_node)]:
                    adj_node_city = graph._node[str(adj_node)]['label']
                    if cur_city != 'n/a' and adj_node_city != 'n/a':
                        if cur_city == adj_node_city:
                            nodes_city[str(node)] += 1

    m = len(graph.edges)
    sum1, sum2, sum3 = 0, 0, 0

    for edge in graph.edges:
        k1 = nodes_city[str(edge[0])]
        k2 = nodes_city[str(edge[1])]
        if k1 == 'n/a' or k2 == 'n/a':
            continue
        else:
            k1 = int(k1)
            k2 = int(k2)
        sum1 += k1 * k2
        sum2 += 1 / 2 * (k1 + k2)
        sum3 += 1 / 2 * (k1 ** 2 + k2 ** 2)
    r_city = (1 / m * sum1 - (1 / m * sum2) ** 2) / (1 / m * sum3 - (1 / m *
        sum2) ** 2)
    print('Assortativity by city', r_city)
    return r_city


def density(graph):
    e = graph.number_of_edges()
    v = graph.number_of_nodes()
    return 2 * e / (v * (v - 1))


class MainWindow(QMainWindow):

    def __init__(self):
        QMainWindow.__init__(self)

        self.setMinimumSize(QSize(800, 400))
        self.setWindowTitle("SGMaker")
```

```
self.setWindowIcon(QIcon('graph.png'))

self.idLabel = QLabel(self)
self.idLabel.setText('Id')
self.idLabel.move(30, 20)

self.densityLabel = QLabel(self)
self.densityLabel.setText('')
self.densityLabel.move(500, 100)
self.densityLabel.resize(300, 32)

self.typeLabel = QLabel(self)
self.typeLabel.setText('Type')
self.typeLabel.move(23, 60)

self.assortativityCityLabel = QLabel(self)
self.assortativityCityLabel.setText('')
self.assortativityCityLabel.move(500, 180)
self.assortativityCityLabel.resize(300, 32)

self.fileLabel = QLabel(self)
self.fileLabel.setText('Save as')
self.fileLabel.move(10, 100)

self.assortativityAgeLabel = QLabel(self)
self.assortativityAgeLabel.setText('')
self.assortativityAgeLabel.move(500, 220)
self.assortativityAgeLabel.resize(300, 32)

self.idField = QLineEdit(self)
self.idField.setText('34314751')
self.idField.move(80, 20)
self.idField.resize(200, 32)

self.clusterCoefLabel = QLabel(self)
self.clusterCoefLabel.setText('')
self.clusterCoefLabel.move(500, 140)
self.clusterCoefLabel.resize(300, 32)

self.nodesLabel = QLabel(self)
self.nodesLabel.setText('')
self.nodesLabel.move(500, 20)
self.nodesLabel.resize(300, 32)

self.edgesLabel = QLabel(self)
self.edgesLabel.setText('')
```

```
        self.edgesLabel.move(500, 60)
        self.edgesLabel.resize(300, 32)

        self.fileField = QLineEdit(self)
        self.fileField.setText('example')
        self.fileField.move(80, 100)
        self.fileField.resize(200, 32)

        self.progressLabel = QLabel(self)
        self.progressLabel.setText('Downloading graph may take some time')
        self.progressLabel.move(20, 180)
        self.progressLabel.resize(450, 32)

        self.fileLabel2 = QLabel(self)
        self.fileLabel2.setText('Open file')
        self.fileLabel2.move(10, 220)

        self.fileField2 = QLineEdit(self)
        self.fileField2.setText('example')
        self.fileField2.move(80, 220)
        self.fileField2.resize(200, 32)

        self.comboBox = QComboBox(self)
        self.comboBox.setGeometry(QRect(80, 60, 200, 32))
        self.comboBox.addItem("Friends")
        self.comboBox.addItem("Friend's friends")
        self.comboBox.addItem("Group")

        collectButton = QPushButton('Download graph', self)
        collectButton.clicked.connect(self.button2_push)
        collectButton.resize(200, 32)
        collectButton.move(80, 140)

        computeButton = QPushButton('Compute metrics', self)
        computeButton.clicked.connect(self.button1_push)
        computeButton.resize(200, 32)
        computeButton.move(80, 260)

    def button2_push(self):

        start_time = time.time()
        self.progressLabel.setText('Collecting in progress')
        try:
            self.fileField2.setText(self.fileField.text())
            download_and_create_graph(#login.textName.text(), login.textPass.
                text(),
```

```
                                    self.comboBox.currentText(), self.idField.text
                                        (), self.fileField.text())

            self.progressLabel.setText('Done!')
        except Exception as e:
            print(traceback.format_exc())
            self.fileField2.setText("")
            self.progressLabel.setText('Invalid data!')
        print("--- %s seconds ---" % (time.time() - start_time))

    def button1_push(self):

        try:
            graph = nx.read_gexf('graphs\\' + self.fileField2.text() + '_city.
                gexf')
            graph2 = nx.read_gexf('graphs\\' + self.fileField2.text() + '_age.
                gexf')
            self.densityLabel.setText('Density = ' + str(density(graph)))
            self.assortativityCityLabel.setText('Assortativity by city = ' +
                str(assortativity_city(graph)))
            self.assortativityAgeLabel.setText('Assortativity by age = ' + str
                (assortativity_age(graph2)))
            self.clusterCoefLabel.setText('Clustering coefficient = ' + str(nx
                .average_clustering(graph)))
            self.nodesLabel.setText('Number of nodes = ' + str(graph.
                number_of_nodes()))
            self.edgesLabel.setText('Number of edges = ' + str(graph.
                number_of_edges()))
        except Exception as e:
            print(traceback.format_exc())
            self.densityLabel.setText('Invalid file name')
            #self.densityLabel.setText(type(e).__name__)
            self.assortativityCityLabel.setText('')
            self.assortativityAgeLabel.setText('')
            self.clusterCoefLabel.setText('')
            self.nodesLabel.setText('')
            self.edgesLabel.setText('')
            pass


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    mainWin = MainWindow()
    mainWin.show()
    sys.exit(app.exec_())
```